VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Libor Burda

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Libor Burda**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Tieto Czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Ing. Tomáš Drábek

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * 1D16 * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 18. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 18. dubna 2016

.....

Tieto Czech s.r.o.
28. října 3346/91
702 00 Ostrava - Moravská Ostrava
IČO 64608051 DIČ CZ64608051

Rád bych na tomto místě poděkoval svým kolegům z týmu za jejich rady a také Mgr. Pavle Dráždilové, Ph.D. za veškerou ochotu a pomoc při vypracovávání této práce.

Abstrakt

Tato práce shrnuje průběh vykonávání individuální odborné praxe ve firmě Tieto Czech, s. r. o. V průběhu odborné praxe musely být prostudovány nástroje a technologie, které pak byly použity pro řešení úkolů z oblasti správy unixových operačních systémů. Dále byla vytvořena aplikace sloužící pro hromadnou kontrolu konfigurace serverů. Řešené úkoly byly přínosné pro rozvoj dovedností a vědomostí v tomto oboru.

Klíčová slova: Unix, Linux, Solaris, HP-UX, AIX, Tieto, Ansible

Abstract

This thesis sums the workflow of individual professional practice in company Tieto Czech, s. r. o. During practice, there were tools and technologies, which had to be learned and were used for solving tasks in field of administration of Unix operating systems. Moreover, application was created to check and compare multiple servers configuration. Solved tasks have been beneficial for development of skills and knowledge in that field.

Key Words: Unix, Linux, Solaris, HP-UX, AIX, Tieto, Ansible

Obsah

Seznam použitých zkratek a symbolů

FS	–	File system
SSH	–	Secure Shell
RHEL	–	Red Hat Enterprise Linux
IBM	–	International Business Machines Corporation
AIX	–	Advanced Interactive eXecutive
HP	–	Hewlett-Packard
HP-UX	–	Hewlett Packard UniX
TONE	–	Tieto Operational Knowledge Engine
OS	–	Operační systém
RAID	–	Redundant Array of Independent Disks

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce je shrnout průběh odborné praxe vykonávané na pozici Specialist, Unix Operations ve firmě Tieto Czech, s. r. o. Tato pozice byla vybrána z důvodu mého dlouholetého zájmu o distribuce operačního systému Linux. Pracovní náplň praxe byla zaměřena výhradně na správu operačních systémů IBM AIX, Oracle Solaris, HP-UX a RedHat Enterprise Linux.

Úvodní část této práce obsahuje charakteristiku společnosti Tieto a její dceřinné firmy Tieto Czech, s. r. o. Dále je uveden popis organizace firmy a harmonogram praxe, ve kterém je stručně shrnut průběh celé praxe.

Druhá část se zabývá popisem používaných operačních systémů a důležitých nástrojů. Na tuto část navazuje rozbor problémů, během praxe řešeny. Je v ní ale také uveden popis a postup řešení úkolu, jehož cílem bylo porovnávání stavů několika serverů před a po jejich restartování.

V závěru jsou uvedeny znalosti získané během praxe a také znalosti, které scházely a bylo třeba je nastudovat. Na úplném konci se nachází zhodnocení celé praxe.

2 Charakteristika firmy Tieto Czech, s.r.o.

Firma Tieto [1] [2] byla založena roku 1968 ve finském městě Espoo pod tehdejším názvem *Tietotehdas Oy*. V roce 1995 byla firma přejmenována na *TT Tieto*. O tři roky později došlo ke změně názvu znovu, tentokrát byl název změněn na Tieto.

Roku 1999 došlo ke spojení s firmou Enator, což vedlo k dalšímu přejmenování na *TietoEnator*. Na nynější název *Tieto Corporation* byla firma přejmenována v roce 2009.

Společnost Tieto je největším poskytovatelem IT služeb v severní Evropě. Firma poskytuje své služby pro zákazníky působící v oblastech lesnictví, telekomunikací nebo také zdravotnictví.

Její hlavní sídlo je ve finských Helsinkách. K dnešnímu dni zaměstnává přes 13000 expertů. V roce 2001 byla založena dceřinná společnost Tieto Czech, s. r. o., což vedlo k rozšíření působnosti firmy do České republiky, kde v současné době zaměstnává přes 2000 lidí.

2.1 Organizace firmy a pracovní zařazení

Firma je organizována na základě metodiky ITIL. Jedná se o souhrn praxí ověřených procesů, úkolů a postupů tak, aby bylo zajištěno lepší plánování a poskytování služeb v oblasti informačních technologií.

Odborná praxe byla vykonávána na pozici Specialist, Unix Operations v ostravské pobočce firmy Tieto. Jedná se o pozici spadající do druhé úrovně podpory v rámci rozdělení podle metodiky ITIL. Náplní práce bylo především řešení incidentů vzniklých na serverech spravovaných naším týmem.

2.2 Harmonogram praxe

Vykonávání mé praxe začalo 1. září 2015. Během praxe byl evidován přibližný čas strávený nad řešením zadaných problémů.

V prvním týdnu bylo nutné absolvovat čtyřdenní vstupní školení, na kterém jsme byli obeznámeni s interními předpisy, zaměřením a chodem firmy. Poté byla zahájena spolupráce s mým týmem.

Ze začátku bylo nezbytné naučit se pracovat s informačním systémem TONE a serverovým monitorovacím nástrojem Patrol od společnosti BMC. Tyto nástroje jsou popsány dále v této publikaci.

Od poloviny září bylo mým úkolem řešit problémy vzniklé na servech, které náš tým spravoval. Tyto problémy byly hlášeny pomocí monitorovacího nástroje Patrol, vyskytovaly se zde ale i manuálně hlášené incidenty.

V průběhu měsíce října proběhlo dvoudenní školení s názvem *Advanced Linux Fundamentals*. Na tomto školení jsme se naučili základní práci s unixovými programy, jako např. AWK, sed, LVM2, ale také konfiguraci linuxového firewallu *iptables* a DNS serveru. Mnoho z těchto znalostí jsem využil později při výkonu praxe.

Celý listopad bylo prováděno vyřazovaní serveru z provozu, jinak označované jako *Server decommission*. Na tomto procesu se podílelo více týmů. Náš tým byl zodpovědný za zjištění konfigurace serveru potřebné k dalším činnostem, jako např. odebrání IP adres z firewallu týmem zodpovědným za provoz sítě. Dále jsme byli zodpovědní za vypnutí serveru a v případě virtualizovaného serveru za smazání virtuálních disků.

V prosinci bylo náplní mé práce mimo jiné řešení problémů se zaplněnými souborovými systémy. Vyřešit tyto problémy vyžadovalo podrobné zkoumání, řešením bylo většinou rozšíření daného souborového systému nebo archivací log souboru. V některých případech bylo nutné přidat fyzický disk. Tento úkon ovšem spadal do kompetence týmu T3.

Od začátku měsíce ledna byla zahájena práce na úkolu, jehož cílem bylo zkontrolovat, zda stav několika serverů po aplikování bezpečnostních záplat a následném restartu odpovídá stavu, ve kterém se servery nacházely před restartem. K tomuto úkolu byl využit nástroj Ansible [5], který je schopen zadaný úkol spustit na několika serverech současně. Řešení tohoto úkolu probíhalo až do poloviny března. Poté byly až do závěru praxe, tj. počátek dubna, opět řešeny incidenty na serverech.

3 Nástroje používané během praxe

Níže v této kapitole jsou popsány operační systémy a důležité nástroje, které byly během praxe používány.

3.1 Operační systémy

Většina dnešních počítačů potřebuje ke svému fungování operační systém. Hlavním úkolem operačního systému je správa hardwarových a softwarových prostředků počítače [3].

Největší zastoupení mezi operačními systémy spravovanými naším týmem patří linuxová distribuce RedHat Enterprise Linux. Za ní následuje operační systém AIX od firmy IBM, Solaris vyvíjený firmou Oracle a HP-UX. Pouze na několika serverech běží distribuce SUSE Enterprise Linux.

Rozdíly mezi těmito operačními systémy jsou značné. Při porovnání implementace najdeme rozdíl nejen v jejich jádrech, ale také v hardwarových architekturách, na kterých jsou tyto operační systémy schopny fungovat.

Všechny uvedené systémy ovšem staví na myšlenkách a filozofických přístupech původního operačního systému UNIX, takže jistou podobnost zde najdeme.

3.2 OpenSSH

Servery byly spravovány vzdáleně, což znamená, že k nim pro naše potřeby nebyl zajištěn fyzický přístup. Pro vzdálenou správu byla využívána aplikace OpenSSH. Jedná se o svobodnou a v dnešní době velice rozšířenou implementaci protokolu SSH.

OpenSSH ke svému fungování potřebuje serverovou část spuštěnou na serveru a klienta, pomocí kterého se z terminálu připojujeme na server. Po připojení se terminál spuštěný na klientském počítači chová stejně, jako bychom pracovali lokálně na vzdáleném serveru.

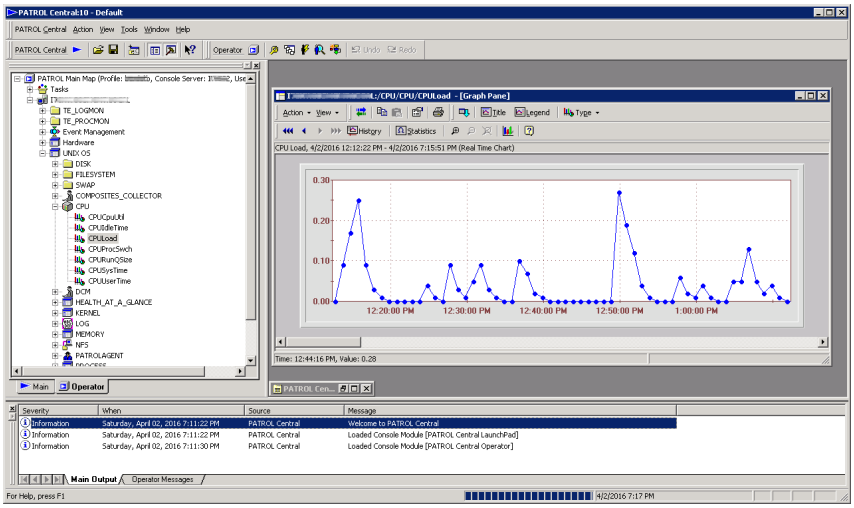
Během komunikace klienta se serverem je veškerý přenos šifrován pomocí šifrovacích algoritmů z rodin AES a RSA [4].

3.3 TONE

Chod téměř celé firmy funguje na principu vytváření a přeposílání tiketů mezi týmy. Aby bylo možné s tikety přehledně a efektivně pracovat, vyvíjí si firma svůj vlastní informační systém zvaný Tieto Operational Knowledge Engine, zkráceně TONE.

Jedná se o velice propracovaný systém založený na webových technologiích. Mezi jeho klíčové vlastnosti patří správa tiketů o incidentech, problémech a změnách. Dále tento systém disponuje databází veškerého hardwaru spravovaného firmou Tieto.

Další důležitou vlastností je úzké propojení s monitorovacím nástrojem BMC Patrol. Díky tomu může monitorovací nástroj vytvářet tikety o incidentech zcela automaticky na základě přednastavených pravidel.



Obrázek 1: BMC Patrol Central

3.4 BMC Patrol

Udržovat obrovské množství serverů v požadovaném funkčním stavu není jednoduché. Proto firma používá nástroj Patrol od společnosti BMC, který je schopný servery v reálném čase monitorovat a v případě incidentu včas uvědomit správce serveru prostřednictvím tiketu.

Architektura nástroje je založena na principu klient-server. Na monitorovaném serveru je spuštěna serverová část zvaná Patrol Agent, jehož úkolem je provádět samotné monitorování. Přístup k naměřených datům je potom možný z aplikace Patrol Console, která umí přehledně zobrazit veškeré sledované parametry monitorovaného serveru.

3.5 BMC BladeLogic

BladeLogic je nástroj sloužící pro automatizace. S jeho pomocí je možné spustit jeden skript na několika vzdálených serverech současně, čímž je možné ušetřit značné množství práce. Pro psaní skriptu se využívá jazyka NSH, který je zpětně kompatibilní se skriptovacím jazykem Bash.

3.6 Ansible

Ansible je, stejně jako BladeLogic, nástroj pro automatizaci správy operačního systému, jeho konfigurace a nasazování aplikací na server [5].

Jeho cílem je poskytnout správcům systémů pohodlnou automatizaci konfigurace serverů, aniž by museli znát některý ze skriptovacích programovacích jazyků nebo řešit veškerou logiku paralelního spouštění úloh. Proto se autoři aplikace rozhodli využít jazyk YML, který je jednoduše čitelný a snadný na naučení, což ho dělá uživatelsky velice přívětivým.

Hlavní výhodou tohoto nástroje je, že ke svému fungování nepotřebuje klienta spuštěného na vzdáleném serveru. Namísto toho využívá standardního klienta OpenSSH, který je dnes

běžnou součástí všech unixových systémů. Nevýhodou tohoto řešení je potřeba distribuce SSH klíče na vzdálené servery.

Automatizace je provedena na základě konfigurace a postupu specifikovaného v tzv. *playbook* skriptu.

```
---
- hosts: webservers
  become: yes

tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
notify:
  - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
handlers:
  - name: restart apache
    service: name=httpd state=restarted
```

Výpis 1: Ukázka playbook skriptu

Na začátku každého skriptu musíme uvést skupinu serverů, na kterých chceme specifikované úkoly spustit. Tyto skupiny musíme specifikovat ve zvláštním souboru hosts v adresáři */etc/ansible*.

Proměnnou *become* říkáme, že chceme veškeré úkoly spouštět jako superuživatel (root). V sekci *tasks* přichází na řadu samotné úkoly spouštěné na cílových serverech. Jednotlivé úkoly jsou spouštěny voláním modulů, které jsou součástí aplikace. V uvedeném příkladu v prvním úkolu nainstalujeme poslední verzi webového serveru Apache, v druhém pak s využitím šablony vytvoříme na vzdáleném serveru jeho konfigurační soubor a ve třetím úkolu se ujistíme, že je Apache spuštěn. U druhého úkolu navíc používáme tzv. *notify*, čímž vynutíme restartování služby Apache.

4 Školení *Advanced Linux Fundamentals*

V říjnu proběhlo dvoudenní školení *Advanced Linux Fundamentals*. Jak už název napovídá, jednalo se o pokročilé školení zaměřené na praktické nastavování linuxového serveru, na kterém běžel operační systém CentOS 7. Spektrum nástrojů , se kterými se na školení pracovalo, bylo široké, proto byly probírány a názorně ukázány pouze podstatné vlastnosti těchto nástrojů.

4.1 Základní unixové nástroje

V úvodu školení byly představeny základní unixové nástroje, mezi které patří především příkaz cat a nástroje AWK nebo sed.

Příkaz cat je poměrně jednoduchý, jeho úkolem je pouze vypsat obsah souboru na standardní výstup, kterým je zpravidla terminál. Výstup je možné přesměrovat pomocí znaků > nebo >> do souboru, respektive na jeho konec.

Utilita AWK slouží pro práci s databázovými daty. Její použití je vhodné především u dat, jejichž struktura je organizována do sloupců. Syntaxe tohoto nástroje se může zdát zprvu poněkud složitá, po jejím zvládnutí máme ale k dispozici velice mocný nástroj.

Sed, neboli *stream editor*, je nástroj sloužící k neinteraktivnímu editování vstupu, který je mu předán. Mezi nejpoužívanější operace prováděné tímto nástrojem patří nahrazování zadaného textu textem jiným, přičemž při zadávání nahrazovaného textu je možné použít regulární výraz.

4.2 RAID

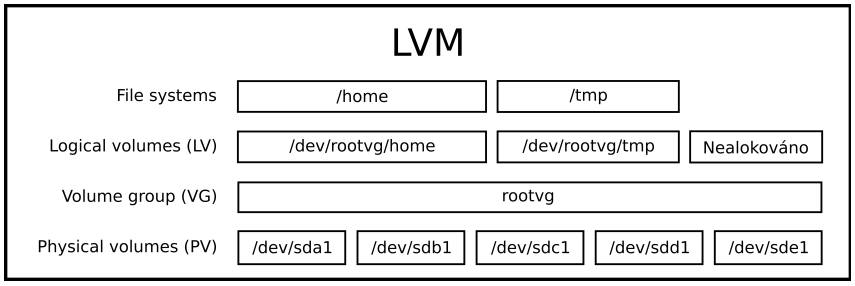
Dále byla probírána technologie RAID a její praktické použití. RAID [6] je způsob spojení dvou a více disků, při kterém se vytváří jeden logický celek za účelem zvýšení výkonu, kapacity nebo odolnosti proti výpadku. Nelze ovšem dosáhnout maxima všech tří vlastností najednou, jedná se spíše o kompromis mezi nimi.

RAID 0, označovaný také jako *striping*, je spojení dvou a více disků do jednoho logického celku, jehož výsledná velikost je rovna součtu kapacit všech disků v poli. RAID 0 ovšem nezvyšuje odolnost dat, při výpadku jednoho disku dochází k degradaci celého diskového pole a nenávratně přicházíme o data na něm uložená.

RAID 1, neboli zrcadlení (angl. mirroring) je spojení disků, při kterém dochází k redundanci dat. Velikost pole je potom rovna velikosti nejmenšího z disků v poli. Tento typ diskového pole zvyšuje odolnost dat, kdy při výpadku jednoho disku nepřicházíme o data.

V praxi se výše uvedené typy diskových polí kombinují, označují se jako RAID 10, který je složený ze dvou RAIDů 1, nad kterými je následné postaven RAID 0. Díky tomu dochází ke zvýšení rychlosti zápisu a zároveň je zaručena redundance dat.

Mezi dnes nejpoužívanější disková pole dále patří RAID 5 a RAID 6. Ty ke svému fungování potřebují tři, respektive 4 disky. RAID 5 funguje v principu tak, že data se ukládají na $n - 1$ disků, kde n je počet disků v poli. Na zbylý disk se ukládá tzv. kontrolní součet, který v případě



Obrázek 2: Diagram znázorňující vrstvy LVM2

selhání disku slouží k dopočítání ztracených dat. RAID 6 se liší pouze v tom, že kontrolní součet ukládá na 2 disky. Díky tomu je toto pole schopné zvládnout výpadek dvou disků současně, aniž by došlo ke ztrátě dat.

4.3 LVM2

LVM je určen výhradně pro OS s jádrem Linux a existuje ve dvou verzích. S verzí linuxového jádra 2.6 nastoupila verze LVM2, která je dnes k dispozici ve většině linuxových distribucí.

LVM slouží především k vytváření a úpravám diskových oddílů aniž by bylo nutné restartovat systém nebo dané disky přepojovat [7].

Principem se LVM podobá diskovému poli RAID 0, který spojuje více fyzických disků do jednoho logického celku. LVM ale navíc poskytuje možnost dané logické celky za běhu zvětšovat, popř. zmenšovat, a vytvářet jejich tzv. snapshoty neboli zachytit a uložit aktuální obsah disku [8].

Koncept LVM je rozdělen do tří vrstev:

- Physical volume, zkráceně PV
- Volume group, zkráceně VG
- Logical volume, zkráceně LV

Aby bylo možno na disku využívat LVM, je nutno nejprve na disku, diskovém oddílu nebo diskovém poli vytvořit PV. K tomuto slouží příkaz *pvcreate*.

Nad vrstvou PV se dále vytváří VG, jejímž úkolem je shlukovat dohromady několik PV. Volume group lze mít v systému několik, k jejich vytvoření slouží příkaz *vgcreate* a je možno je rozšířit pomocí *vgextend*.

Poslední vrstvou je vrstva LV. Na této vrstvě se vytváří logické LVM oddíly, které jsou v systému prezentovány jako běžné logické oddíly vytvořené na fyzických discích. Velikost jedné LV je omezena velikostí VG, nad kterou se LV vytváří.

4.4 iptables

iptables je nástroj příkazového řádku a v Linuxu plní funkci paketového filtru. iptables [12] je postaven na principu řetězů (z angl. chains), kdy každý paket přijatý nebo odeslaný z dané stanice musí projít alespoň jedním z řetězů.

Ve výchozím nastavení obsahuje iptables řetězy tři: INPUT, OUTPUT a FORWARD. V každém z těchto řetězů jsou specifikována pravidla, na která se zpracovávaný paket testuje. Paket prochází pravidly postupně odshora dolů. Pokud paket danému pravidlu vyhoví, provede se akce specifikovaná v daném pravidle a dále se v testování nepokračuje.

Na konci každého řetězu je nastavena výchozí politika, která říká jak má být s paketem naloženo, pokud není splněna žádná z podmínek. V takovém případě může být paket buď zahozen nebo propuštěn dále do systému.

5 Praktické úkoly

Tato kapitola obsahuje popis praktických úkolů vypracovávaných během praxe a postup jejich řešení.

5.1 Zaplněný filesystem

Data na serverech jsou ukládána do persistentní paměti. Ve většině případů se k tomu využívají pevné disky, ať už lokální, síťové nebo RAIDy s podporou redundance. Jelikož je kapacita fyzického úložiště omezena, často se stává, že se toto úložiště po nějakém čase zaplní. Řešení takového problému je několik.

5.1.1 Nedostatek místa v kořenovém adresáři

Běžným problémem je nedostatek místa na fyzických discích připojených do kořenového adresáře. V unixových operačních systémech má souborový systém stromovou strukturu, na jejímž počátku je kořen (v angličtině označován jako *root*).

V adresáři root se pak nacházejí různé systémové adresáře, které mohou být uloženy na jednom diskovém oddílu nebo se mohou nacházet dokonce i na různých fyzických discích, které jsou do této stromové struktury připojeny.

Výhoda rozložení systémových adresářů na více oddílů spočívá v tom, že lze kontrolovat velikost jednotlivých adresářů pomocí omezení velikosti diskového oddílu. Pokud tedy máme jeden adresář na jiném oddílu a tento adresář se nečekaně začne plnit daty, zaplní se pouze daný diskový oddíl čímž není ohroženo fungování zbytku systému. V případě, že bychom měli všechny adresáře na jednom oddílu, zaplnil by se celý oddíl a na disk by v danou chvíli nemohl zapisovat vůbec žádný proces a může dojít dokonce k havárii celého systému.

V případě zaplnění filesystému bylo nejprve nutné prozkoumat, čím je zaplnění způsobeno. K tomu následující příkaz:

```
# du -a /var | sort -n -r | head -n 10
```

Výpis 2: Nalezení 10 největších souborů

Nejčastěji zaplnění FS způsoboval log soubor. Pokud se jednalo o log soubor patřící operačnímu systému (např. soubor */var/log/messages* pro OS Linux), log soubor se musel prostudovat, jelikož se zde pravděpodobně neustále zapisuje nějaká systémová chyba. Pokud v log souboru nebyla nalezena žádná chyba, log soubor byl odrotován.

V případě adresáře */home*, který slouží k ukládání dat uživatelů, musel být filesystém připojený do tohoto adresáře rozšířen pomocí LVM, aby nedošlo k úplnému zaplnění, a následně byli uživatelé prostřednictvím e-mailu informováni, aby uvolnili místo ve svých domovských adresářích. Žádný systémový administrátor by totiž neměl zasahovat do soukromých souborů uživatelů.

5.1.2 Nedostatek místa v oddílu pro swap

Swap je speciální druh souborového systému používaný v unixových systémech. Slouží jako odkládací prostor pro data, pro která již není místo v operační paměti. Pokud je tedy operační paměť do určité hodnoty (tuto hodnotu lze nastavit pomocí systémové proměnné *vm.swappiness*) zaplněna, dochází k tzv. swapování, tedy přesunu stránek z operační paměti do odkládacího prostoru systému.

Odložené stránky bývají přesunuty zpět až v momentě, když si k nim vyžádá přístup právě běžící proces, kterému tyto stránky patří. Přesun stránek z odkládacího prostoru do paměti RAM je časově náročný a způsobuje také zpomalení dotyčné aplikace. Proto je vhodné všechny stránky z odkládacího prostoru přesunout do paměti RAM. Předpokládá se ovšem, že je v paměti RAM dostatek volného prostoru.

Abychom tento problém vyřešili, musíme na serveru spustit následující příkaz, který zašle jádru OS pokyn uvolnění tzv. pagecache, která slouží jako vyrovnávací paměť.

```
# echo 1 > /proc/sys/vm/drop_caches
```

Výpis 3: Uvolnění pagecache

Prvním z následujících příkazů poté zjistíme, zda je v paměti RAM dostatek místa. Pokud ano, spustíme druhý příkaz, který zakáže používání všech zařízení, která slouží jako odkládací prostor, a zároveň data z těchto zařízení přesune do paměti RAM. Tato operace může trvat v závislosti na velikosti odkládacího prostoru až jednu hodinu. Nakonec je nutné používání odkládacího prostoru opět povolit. To provedeme posledním příkazem.

```
# free -mo
# swapoff -a
# swapon -a
```

Výpis 4: Uvolnění odkládacího prostoru

Tím jsme uvolnili místo v odkládacím prostoru. Nutno podotknout, že tento proces sloužil pouze jako operativní opatření. Pokud k zaplňování odkládacího prostoru docházelo příliš často, musel být tento problém řešen se zkušenějšími kolegy, kteří poté tento problém řešili buď přidáním paměti RAM nebo kontaktovali tým zodpovědný za běžící aplikaci.

5.2 Vytváření uživatelských účtů

Unixové operační systému jsou víceuživatelské, což umožňuje práci několika uživatelů současně. K tomu je nutné, aby měl každý uživatel svůj systémový účet, kterým se do systému autorizuje.

Uživatelské účty byly vytvářeny na základě *Service requestu*, tedy požadavku, který je speciálním případem *Incident ticketu*. Často bylo požadováno založit uživatelský účet na více serverech. K tomu bylo vhodné využít nástroje BladeLogic a tím tento úkon automatizovat.

5.2.1 Řešení

Pro vytváření použijeme následující skript, který má za úkol vytvořit účet a nastavit u něj počáteční heslo a dobu, po kterou bude možné účet používat.

```
#!/bin/bash

username="Petr Novak"
login="novakpetr"
groups="users,wheel,sysadmin"

returnvalue=false
getent passwd $login >/dev/null 2>&1 && returnvalue=true

if $returnvalue; then
    exit
fi

useradd -c "System Administrator / $username" -m -G $groups $login
echo "P@ssword123" | passwd $login --stdin
chage -I -1 -m 0 -M 99999 -E -1 $login
```

Výpis 5: Skript pro vytváření uživatelských účtů

Nejprve musíme do skriptu doplnit potřebné náležitosti jako jsou přihlašovací jméno, skutečné jméno uživatele a skupiny, do kterých bude uživatelský účet zařazen. Tyto údaje získáme z odpovídajícího požadavku v nástroji TONE.

Následně spustíme skript v nástroji BladeLogic, kde ze seznamu serverů vybereme pouze servery, na kterých chceme účet vytvořit.

Na servech s OS AIX byly spuštěny databáze DB2, kde pro autorizaci do systému i do databáze slouží jeden účet. Proto bylo nutné nastavovat výchozí heslo a jeho atributy, aby heslo nikdy nevypršelo. Uživatelé se totiž přihlašovali přímo z databázového klienta, kde není možné heslo měnit po prvním přihlášení ani po jeho expiraci.

5.3 Server decommission

Server decommission je proces, při kterém dochází k vyřazení serveru z provozu, ať už z důvodu jeho nepotřebnosti nebo například zastarání daného stroje.

Celý proces se skládá z přibližně 10 úkolů (tasků) v závislosti na tom, zda je server fyzický nebo virtualizovaný.

Naše práce začíná v momentě, kdy dostaneme vytvořený tiket o provedení změny, ve kterém vyplníme nezbytné náležitosti, jako předpokládané datum ukončení procesu. Poté musí být tiket schválen manažerem zodpovědným za změny a je vygenerována první část úkolů, která je automaticky přiřazena na odpovídající fronty.

Když je první část úkolů hotová, přichází na řadu úkol označený *150 - Server shutdown*. Zde je naším úkolem získat několik potřebných údajů a následně daný server vypnout.

```
# dmidecode | grep -i "Serial Number" | grep -v Not | head -1
# systool -c fc_host -v | grep port_name | cut -d '"' -f 2
# multipath -ll
# pvs
# df -h
# ip a
# ip r
```

Výpis 6: Příkazy pro proces *Server decommission*

Tyto příkazy jsou poměrně jednoduché, první zjišťuje sériové číslo serveru pomocí utility *dmidecode*, dále se zjišťují čísla WWPN, které identifikují síťové disky připojené k serveru, příkaz *pvs* zjišťuje fyzické disky, které jsou součástí LVM. Nakonec se vypisují IP adresy a cesty.

Pokud máme všechny informace, můžeme přistoupit k úplnému vypnutí serveru.

Tím je tento úkol hotov, po jeho zavření se vygenerují zbývající úkoly, do kterých musíme doplnit informace získané v úkolu předešlém, a přiřazujeme je na odpovídající fronty.

V případě virtuálního serveru musíme u posledního úkolu v celém procesu ještě odstranit virtuální disky. Tím tento proces končí.

5.4 Vyprázdnění log souborů

Dnešní operační systémy zaznamenávají různé více či méně důležité události do tzv. log souborů. Jak události časem přibývají, log soubory se plní a je třeba je tzv. odrotovat. Proces spočívá v archivaci log souboru a jeho následného vyprázdnění.

Celý úkon musí být proveden za běhu, a to za běhu serveru i aplikace. Na některých linuxových serverech je k tomuto úkonu k dispozici nástroj *logrotate*, který je spouštěn nástrojem Cron. Ten slouží k pravidelnému opakovanému spouštění zadaných úloh.

Tento nástroj ale není nainstalován a nakonfigurován všude, proto se tato akce musí provádět manuálně.

```
# FILE=asa.log
# cat $FILE | gzip -v9 > /tmp/$FILE.`date +%y%m%d`.gz \
> && cat /dev/null > $FILE \
> && mv /tmp/$FILE.$(date +%y%m%d).gz ./
```

Výpis 7: Odrotování log souboru

Na začátku je proměnná FILE nastavena na hodnotu názvu logu. Obsah daného souboru je poté předán utilitě *gzip*, která obsah souboru zkomprimuje a uloží dočasně do složky */tmp*. Do původního souboru se poté přesměruje obsah speciálního zařízení */dev/null*, čímž je zajištěno vyprázdnění původního souboru [9]. Nakonec je proveden přesun zkomprimovaného log souboru pojmenovaného aktuálním datem z */tmp* zpět do složky, ve které se nacházel původní log soubor. Tento skript je schopen fungovat na všech platformách využívajících *Bash* shell.

Vytváření komprimovaného souboru ve filesystému */tmp* má dobrý důvod. Většinou je totiž filesystém, ve kterém je uložen log soubor, téměř úplně zaplněný, není zde tedy možné vytvářet jakékoliv další soubory.

5.5 Kontrola konfigurace serveru po restartu

Na začátku ledna byla s Tomášem Novobilským zahájena spolupráce na úkolu, jehož cílem bylo vyvinout jednoduchou aplikaci s využitím nástroje Ansible. Úkolem této aplikace bylo porovnat konfigurace několika serverů před a po jejich restartu, který byl často vyžadován po instalaci bezpečnostních záplat. Kontrolu konfigurace byla dosud prováděna manuálně s využitím nástroje *conf2html*, který je schopen uložit do souboru konfiguraci serveru ve formátu HTML. Soubory získané před a po restartu serveru pak byly mezi sebou porovnány pomocí jednoúčelových utilit. Toto řešení ovšem není vhodné, protože se tímto způsobem musely porovnávat výstupy jeden po druhém.

5.5.1 Testovací prostředí

Aplikace byla vyvíjena v prostředí využívající virtualizační technologii VMWare VCenter. Zde bylo virtualizováno pět serverů s operačním systémem CentOS, z nichž jeden sloužil jako hlavní (master) server a zbývající čtyři jako vzdáleně spravované servery (tzv. slave servery). Aby testování odpovídalo co nejpřesněji reálným podmínkám, byly na vzdálených serverech nainstalovány různé verze operačního systému CentOS, a to verze 6, 6.2, 6.6 a 7.

5.5.2 Analýza úkolu

Před začátkem samotného vývoje bylo nutné zjistit požadavky na aplikaci. Teprve poté bylo možné přistoupit k vývoji aplikace.

Hlavními požadavky byl především minimální počet závislostí na vzdálených serverech. Dalším požadavkem byl výstup aplikace, kterým měl být pouze jeden soubor v jazyce HTML, který měl obsahovat přehledné výsledky porovnání konfigurací všech serverů.

5.5.3 Implementace

K získání konfigurace serveru byl napsán skript v jazyce Bash, který byl následně pomocí nástroje Ansible spuštěn na vzdálených serverech. Výstup tohoto skriptu byl uložen do souboru v adresáři */tmp*. Pro další porovnání bylo nutné zajistit odpovídající strukturu obsahu souboru.

K tomu byl zvolen jazyk JSON (JavaScript Object Notation). Jedná se o jazyk určený k ukládání dat v textové podobě s velice jednoduchou syntaxí. Podpora tohoto jazyka je implementována pomocí knihovny napsané v jazyce Python, který bude využit při pozdějším porovnávání.

Výstupní soubor byl poté z adresářů */tmp* na vzdálených serverech překopírován pomocí Ansible na hlavní server a uložen do adresářů pojmenovaných podle obsahu proměnné *HOSTNAME*.

Poté byly na serverech administrátorem nainstalovány bezpečnostní aktualizace a server byl restartován. Tyto akce musely být provedeny manuálně z důvodu možných chyb, které při instalaci mohly nastat.

Po úspěšné instalaci aktualizací byl znovu spuštěn skript pro získání konfigurace a výstup uložen do odpovídajících adresářů.

Posledním krokem v tomto procesu bylo spuštění samotné aplikace pro porovnání konfigurací. K její implementaci byl využit skriptovací jazyk Python. Důvodem pro zvolení právě tohoto jazyka byla zejména přítomnost knihovny *difflib*, která slouží k porovnávání dvou řetězců znaků. Další užitečnou knihovnou se ukázala být knihovna *Jinja2*, která implementuje rozhraní pro snadnou práci s HTML šablonami.

Výsledná aplikace byla implementována tak, že byl nejprve načten obsah adresářů a bylo zkontrolováno, zda některý ze souborů neschází. Následně bylo možné načíst obsahy souborů pro každý server do datových struktur pomocí knihovny JSON.

Získané struktury byly mezi sebou porovnány funkcí ndiff implementované v knihovně difflib. Výstupem porovnání byla třetí datová struktura, ve které byly zaznamenány rozdíly struktury druhé oproti struktuře první.

Ve finální fázi byly výsledky porovnání předány funkci, která zajistila jejich integraci do HTML šablony napsané v jazyce Jinja2. Výsledná HTML stránka byla uložena do adresáře, ze kterého byla přístupná přes webové rozhraní.

6 Závěr

Odborná praxe byla velice přínosná především díky možnosti vyzkoušet si práci s více operačními systémy z rodiny Unix. Využít plný potenciál těchto operačních systémů je totiž většinou možné pouze ve firemní sféře a jen těžko se toho dosahuje v domácích podmínkách.

Dalším přínosem bylo bezpochyby vyzkoušet si práci v týmu a poznat chod a fungování nadnárodní korporace. V průběhu praxe bylo také nutností aktivně využívat anglický jazyk, ať už slovem či písmem.

V průběhu praxe byly využity znalosti z předmětů Operační systémy a Správa operačních systémů. První z těchto předmětů se zaměřuje na principy fungovaní operačních systémů obecně, druhý zmíněný pak na správu operačních systémů, konkrétně distribuce Debian. Z předmětů zabývajících se programováním byla využita znalost skriptovacího jazyka Python vyučovaného v předmětu Skriptovací programovací jazyky.

Mnoho znalostí ovšem scházelo a bylo potřeba je nastudovat. Jednalo se zejména o základy operačních systémů IBM AIX, Oracle Solaris a HP-UX a pokročilou znalost jazyka Bash, který je v těchto systémech využíván. Dále bylo potřeba seznámit se s firemními nástroji a nástrojem Ansible, který byl použit při vývoji aplikace pro hromadnou kontrolu nastavení serverů.

Subjektivně hodnotím praxi jako velice přínosnou jak pro mě tak pro firmu, se kterou bude spolupráce v budoucnu pokračovat.

Literatura

[1] *Tieto Czech zahájilo poskytování IT služeb pro finskou vládu* [online]. 2015 [cit. 2016-01-18]. Dostupné z: http://www.tieto.cz/zpravy/tieto-czech-zahajilo-poskytovani-sluzeb-pro-finskou-vladu

[2] *Informace o Tieto* [online]. 2015 [cit. 2016-01-21]. Dostupné z: http://www.tieto.cz/tieto-o-nas

[3] TANENBAUM, Andrew S a Albert S WOODHULL. *Operating systems: design and implementation.* 3rd ed. Upper Saddle River, N.J.: Pearson Prentice Hall, 2006. ISBN 01-314-2938-8.

[4] *OpenSSH Features. OpenSSH* [online]. 2016 [cit. 2016-03-26]. Dostupné z: http://www.openssh.com/features.html

[5] *OVERVIEW: How Ansible Works. Ansible.com* [online]. 2015 [cit. 2016-01-20]. Dostupné z: https://www.ansible.com/how-ansible-works

[6] *Správa linuxového serveru: RAID teoreticky. Linux E X P R E S* [online]. 2009 [cit. 2016-03-26]. Dostupné z: http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-raid-teoreticky

[7] *LVM2 - dynamické vytváření diskových oddílů. AbcLinuxu* [online]. 2015 [cit. 2016-02-10]. Dostupné z: http://www.abclinuxu.cz/clanky/system/lvm2-dynamicke-vytvareni-diskovych-oddilu

[8] *LVM - 1 (úvod, vytvoření oddílu). AbcLinuxu* [online]. 2009 [cit. 2016-02-10]. Dostupné z: http://www.abclinuxu.cz/clanky/system/lvm-1-uvod-vytvoreni-oddilu

[9] *Clear a file using /dev/null in Linux. Programming in Linux* [online]. 2015 [cit. 2016-02-19]. Dostupné z: https://linuxprograms.wordpress.com/2010/05/14/clear-a-file-dev-null/ https://linuxprograms.wordpress.com/2010/05/14/clear-a-file-dev-null/

[10] *Logrotate: zkrocení zlých logů. Root.cz* [online]. 2012 [cit. 2016-03-26]. Dostupné z: http://www.root.cz/clanky/logrotate-zkroceni-zlych-logu/

[11] ERIC STEVEN RAYMOND. *The art of UNIX programming: [with contributions from thirteen UNIX pioneers, including its inventor, Ken Thompson].* [Nachdr.]. Boston [u.a.]: Addison-Wesley, 2004. ISBN 01-314-2901-9.

[12] *Správa linuxového serveru: Linuxový firewall, základy iptables. Linux E X P R E S* [online]. 2010 [cit. 2016-03-26]. Dostupné z: http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-linuxovy-firewall-zaklady-iptables